COMPUTER PROGRAMMING

(Applied Technology Education)

Standards and Objectives

A Project of the Utah State Office of Education

Steven O. Laing, Ed.D.
State Superintendent of Public Instruction

Patti Harrington,
Associate Superintendent
Curriculum and Instructional Services

Mary Shumway
State Director
Applied Technology Education

Parker K. (Duke) Mossman
Education Specialist
Information Technology

December 2, 2003 - Draft

UTAH STATE BOARD OF EDUCATION

250 East 500 South
Salt Lake City, UT 84111

District 1
Max L. Torres
1414 East 1800 South
St. George, UT 84790
Phone: (435) 628-5031

District 2
A. Earl McCain
5762 West Wasatch
Morgan, UT 84050
Phone: (801) 876-3282

District 3
Marilyn Shields
458 Country Club
Stansbury Park, UT 84074
Phone: (435) 882-7137

District 4
Teresa L. Theurer
66 Canterbury Circle
Logan, UT 84321
Phone: (801) 753-0740

District 5
Greg W. Haws
5841 West 4600 South
Hooper, UT 84315
Phone: (801) 985-7980

District 6
Joyce W. Richards
5273 South Winchester Lane
Ogden, UT 84403
Phone: (801) 479-5370

District 7
Kim R. Burningham
932 Canyon Crest Drive
Bountiful, UT 84010
Phone: (801) 292-9261

District 8
John C. Pingree
1389 Harvard Avenue
Salt Lake City, UT 84105
Phone: (801) 582-5635

District 9
Judy Larson
5058 West Corilyn Circle
West Valley City, UT 84120
Phone: (801) 969-2382

District 10
Denis R. Morrill
6024 South 2200 West
Salt Lake City, UT 84118
Phone: (801) 969-2334

District 11
David L. Moss
1964 Hawk Circle
Sandy, UT 84092
Phone: (801) 572-6144

District 12
Laurel Brown
5311 South Lucky Clover Ln
Murray, UT 84123
Phone: (801) 261-4221

District 13
Janet A. Cannon
5256 Holladay Blvd.
Salt Lake City, UT 84117
Phone: (801) 272-3516

District 14
Mike Anderson
455 East 200 North
Lindon, UT 84042
Phone: (801) 785-1212

District 15
Linnea S. Barney
1965 South Main Street
Orem, UT 84058
Phone: (801) 225-4149

Board of Regents Appointment
Pamela J. Atkinson
Intermountain Health Care
36 South State
Salt Lake City, UT 84111
Phone: (801) 442-3511

Board of Regents Appointment
David J. Jordan
201 South Main, #1100
Salt Lake City, UT 84111
Phone: (801) 578-6919

Steven O. Laing
Executive Officer

Twila B. Affleck
Secretary

Foreword

In 1987, the Weber School District initiated a program to address a growing job demand for programming technicians in the Ogden/Weber region. Gradually it evolved into a course that became available statewide. This curriculum outline is an overview of best practices from teachers who have placed students in meaningful employment, internships, concurrent enrollment programs, etc., after students satisfactorily met basic competencies in elementary programming.

This curriculum guide is designed only to meet the goals of Applied Technology Education (ATE) and is not founded on any specific hardware or software requirements. The purpose of this curriculum guide is to provide structure for the delivery of basic Core skills and knowledge in computer programming. This standardization of curricula provides a consistent format for evaluation and certification of students' skills, while allowing programs to access a wide variety of delivery strategies and resources.

This Computer Programming curriculum consists of two one-year courses: Computer Programming I, and Computer Programming II. Students who have time in their schedule for a third year of programming should work with their teacher to design an independent project focusing on advanced programming concepts or languages that the student would like to specialize in. C++, Visual Basic (VB) and Java are the programming languages supported by this curriculum.

For further assistance with issues related to ATE programs in computer programming, contact the Applied Technology Education Division of the Utah State Office of Education at (801) 538-7840.

Acknowledgments

The Utah State Office of Education wishes to acknowledge the efforts of the following individuals in designing and developing this curriculum outline:

Duke Mossman, Project Leader
Utah State Office of Education

Teachers who have contributed to this project
Kelly Albers, Ogden-Weber Applied Technology College
Stacie Bateman, Layton High School
Paul Cohee, Jordan High School
Don Cooley, Utah State University
Matthew Flatt, University of Utah
Shanon Johnson, Uintah High School
Jamie Kreyling, Snow Canyon High School
Art Lee, University of Utah
Kirk Love, Utah Valley State College
Gordon Moses, Pleasant Grove High School
Kim Murphy, Weber High School
Dale Slade, Roy High School
Bob Stratton, Timpanogos High School
Natalie Watts, Taylorsville High School
Troy Young, Snow College South

Others who have contributed to this project
Kal Bumgardner, SOSytems
Scott Lemon, HumanXtensions
Steve Royce, Church of Jesus Christ of Latter-Day Saints (formerly of Orem High School)

Special thanks to all those who have offered their insights, expertise, and assistance in this project.

Table of Contents

Introduction to
Computer Programming

Computer Programming is a sequence of courses that teaches students critical thinking, problem solving and teamwork skills through the use of industry standard programming languages. Those who complete the two-year program will qualify for entry-level positions in the programming/software engineering industry and be prepared for related occupational and educational goals in higher education. The courses reflect a hands-on, project-based series in which students learn the process of producing computer application programs. The curriculum is language-independent but will require the use of at least one current computer programming language (C++, Visual Basic or Java) that supports the objectives and standards presented in this course description. The individual school will provide each student access to a workstation at which they enter and test their programs.

Course Mission/Philosophy

Computer Programming is a course of study that teaches students critical thinking, problem solving and teamwork skills through the application of programming/engineering methodology. It enables students to produce computer programs through the use of industry standard computer languages. The course also qualifies students for entry-level industry positions and prepares them in achieving other occupational/educational goals at the college and university level. The courses are also intended to expose students to the computer programming/software engineering industry through work-based learning experiences to help them decide on career path and chart a course to obtain their goal through the Student Education Occupation Plan (SEOP) process.

Suggested Course Descriptions and Delivery Model

Year 1

Computer Programming I/A - Semester. An introduction to computer programming/software engineering and applications. The course introduces students to the fundamentals of computer programming, to simple control and data structures, to basic operating system commands, and to the use of text files. Students will learn to design, code, and test their own programs. It is recommended that teachers use the Scheme system for teaching this first semester of computer Programming. A skill certification exam is not available for this one semester course - see below.

Suggested prerequisite: Completion of, or concurrent enrollment in, algebra I, keyboarding proficiency, and computer technology.

Computer Programming I/B - Semester. An intermediate class in computer programming/software engineering and applications. Reviews and builds on the concepts introduced in CEI-A. Introduces students to more complex data structures and their uses, including sequential files, arrays, classes, and recursive processes. Students will learn to create more powerful programs.

Suggested prerequisite: Successful completion of CEI/A.

Note: Computer Programming I/A and Computer Programming I/B can be combined and offered as a full year course. Skill certification is only available at the end of Computer Programming I/B.

Year 2

Computer Programming II - Full year. An advanced class in computer programming/software engineering and applications. Reviews and builds on the concepts introduced in CEI-B. Introduces students to dynamic allocation of data, advanced utilization of classes, advanced GUI techniques, and advanced applications of recursion.

Suggested prerequisite: Successful completion of CEI/B.

Year 3

Computer Programming Individual Projects - Full Year. (Use Computer Programming II CIP Code). An advanced class in computer programming/software engineering where students can concentrate on their area of interest as directed by the instructor.

Computer Programming I

Grade Levels: 10-12
Units of Credit: 1
CIP Code: 11.0201
Prerequisite: Completion of, or concurrent enrollment in, Algebra I, Keyboarding Proficiency, and Computer Technology (Computer Literary)
Skill Certification Exam: #802

COURSE DESCRIPTION

A beginning through intermediate course in computer programming/software engineering and applications. The course introduces students to the fundamentals of computer programming, simple control and data structures, basic operating system commands, sequential files, arrays, classes, recursive processes, and the use of text files. Students will learn to design, code, and test their own programs. It is recommended that teachers use the Scheme system to introduce programming concepts and problem solving skills to beginning programming students, followed by an introduction to C++, Visual Basic or Java.

COURSE STANDARDS AND OBJECTIVES

STANDARD
110201-01 Students will be familiar with and use a programming environment.

OBJECTIVES

110201-0101 Demonstrate knowledge of external and internal computer hardware.

- Describe the functions of basic computer hardware devices (monitor, system board, printer, CD-ROM drive, hard drive, floppy drive, keyboard, mouse, adapters, ports, other devices).
- Describe the functions of the internal components of computers (CPU, RAM, ROM, registers, ALL).
- Translate to and from binary code (Computer number systems).

110201-0102 Demonstrate knowledge of software concepts.

- Define computer software.
- Explain the process of software installation.

110201-0103 Develop the ability to use a current operating system.

- Demonstrate how to load, save and back up files.
- Demonstrate how to rename and delete files.
- Demonstrate how to move, copy and compress files.
- Demonstrate how to display and print files.

- Demonstrate the ability to manage files on a PC and network.
  - Create Folders
  - Create and use appropriate directory and path structures
  - Copy files between folders
  - Understand the organization of files on a hard drive and a network
- Demonstrate how to execute a program.

110201-0104 Demonstrate the ability to use the editor to enter programs.
- Demonstrate how to enter text and commands.
- Demonstrate the process of selecting a block of text.
- Demonstrate how to move blocks of text.
- Demonstrate how to copy blocks of text.
- Demonstrate how to delete blocks of text.

110201-0105 Demonstrate the ability to compile, debug and execute programs.
- Demonstrate how to use the editor to compile and run programs.
- Understand the difference between syntax, run-time, and login errors.
- Demonstrate how to debug programs.

STANDARD
110201-02 Students will employ accepted programming methodology.

OBJECTIVES;
110201-0201 Demonstrate the ability to use good programming style.
- Demonstrate how to use white space properly.
- Employ the use of case-sensitive commands for clarity.
- Construct programs with meaningful identifiers.

110201-0202 Employ the proper steps to programming in order.
- Prepare specifications for computer programs.
- Design solutions using computer programs.
- Develop the code for a program.
- Test programs for effectiveness and completeness.
- Provide full documentation for a program.

110201-0203 Employ proper program design process.
- Use step-wise refinement (top-down design) in programming.
- Employ program modularity in writing programs.
- Produce logical algorithms to solve problems with a computer program.

110201-0204 Demonstrate the ability to program for automatic error checking (robustness).
- Explain how to protect program execution from incorrect input.
- Describe how to protect program execution from run-time errors.

- Employ verification to protect program results from logic errors.

STANDARD
110201-03 Students will properly use language-fundamental commands and operations.

OBJECTIVES:

110201-0301 Demonstrate the ability to use basic elements of a specific language.
- Write programs using a language-specific template.
- Declare and assign values to constants and variables in programs.
- Employ arithmetic expression in programs.
- Output text with formatting.
- Demonstrate how to pause programs in order to view the output.
- Demonstrate the ability to use input/output commands in programs.
- Input values into identifiers.
- Output values stored in identifiers.

STANDARD
110201-04 Students will properly employ control structures.

OBJECTIVES:

110201-0401 Demonstrate the ability to use relational and logical operators in programs.
- Compare values using relational operators.
- Form complex expressions using logical operators.
- Demonstrate how to use operator overloading (C++ and Java).

110201-0402 Demonstrate the ability to use decisions in programs.
- Employ simple IF structures.
- Use IF-ELSE structures.
- Write programs with nested IF-ELSE structures.
- Make multiple-way selections (switch, case).

110201-0403 Demonstrate the ability to use loops in programs.
- Use initial, terminal, and incremental values in loops.
- Construct both pre-test and post-test loops.
- Demonstrate how to use counted loops.
- Describe the use of flagged (sentinel-controlled) loops.
- Utilize nested loops.
- Explain how to avoid infinite loops.
- Accumulate running totals using loops.

110201-0404 Demonstrate the ability to use recursion in programs.
- Create a recursive process.
- Explain how to implement recursion.

- Evaluate a recursive process.

110201-0405 Demonstrate the ability to use sub-programs in programs.
- Demonstrate how to use language-predefined sub-programs.
- Call sub-programs.
- Develop sub-programs.
- Send to and retrieve data from sub-programs.
- Utilize value, constant, and reference parameters.
- Understand the scope of identifiers in sub-programs.

STANDARD
110201-05 Students will employ proper static data structures.

OBJECTIVES:

110201-0501 Demonstrate the ability to use atomic data types in programs.
- Declare and use integer identifiers.
- Declare and use character identifiers.
- Declare and use floating point (real) identifiers.
- Declare and use Boolean identifiers.
- Declare and use constants.

110201-0502 Demonstrate the ability to use static arrays in programs.
- Declare arrays all applicable types.
- Initialize arrays.
- Input data into arrays.
- Output data from arrays.
- Perform operations on arrays.
- Perform sequential searches on arrays.
- Perform a quadratic sort on an array.
- Utilize multidimensional arrays.

110201-0503 Demonstrate the ability to use strings in programs.
- Declare string identifier.
- Input string identifiers.
- Output string identifiers.
- Compare string identifiers.
- Find the length of a string.
- Copy part or all of string identifiers into other strings.
- Concatenate string identifiers.
- Locate and delete sub-string positions.
- Insert strings into other strings.

STANDARD
110201-06 Students will properly employ object-oriented programming techniques.

OBJECTIVES;
110201-0601 Demonstrate the ability to use classes.
- Use objects.
- Use object data members.
- Use object member functions (methods).

110201-0602 Demonstrate the ability to create and use user-defined classes.
- Create and use user-defined data members.
- Create and use user-defined methods.

STANDARD
110201-07 Students will properly use sequential files.

OBJECTIVES:
110201-0701 Demonstrate the ability to use sequential files in programs.
- Create and initialize sequential files.
- Store data to sequential files.
- Retrieve data from sequential files.
- Update sequential files.

STANDARD
110201-08 Students will apply appropriate programming skill as an effective member of a team.

OBJECTIVES;
110201-0801 Demonstrate the ability to apply knowledge to a programming project.
- Formalize specifications.
- Choose proper input parameters.
- Choose appropriate data structures and processing.
- Design appropriate output.
- Use appropriate test data.
- Write good documentation.

110201-0802 Demonstrate the ability to use teamwork and collaboration in a programming project.
- Divide a project among programmers.
- Present work to a group.
- Coordinate work with others in the group.

- Complete assigned work according to predetermined deadlines.
- Participate in a peer performance evaluation.
- Demonstrate professionalism in team relationships, communication, timeliness, and attitude.

STANDARD
110201-09 Students will demonstrate knowledge of current ethical issues dealing with computers and information in society.

OBJECTIVES;

110201-0901 Demonstrate knowledge of programmer ethics.
- Show knowledge of the importance of correct programming.
- Check for program correctness using verification.
- Know proper and improper standards in programming.

110201-0902 Demonstrate knowledge of the social and ethical consequences of computers.
- Describe how computer-controlled automation affects workers and management.
- Explain the ramifications of society's dependance on computers.
- Identify advantages and disadvantages of changing workplace environments.

110201-0903 Demonstrate knowledge of the right to privacy.
- Explain how computers can compromise privacy.
- Exhibit knowledge of privacy laws.
- Describe responsibilities of people who control computer information.

110201-0904 Demonstrate knowledge of computer, information and software security.
- Exhibit knowledge of copyright laws.
- Explain how computers can be used to compromise copyright laws.
- Give examples of ways to protect information on computer systems.
- Identify ways to protect against computer viruses.

STANDARD
110201-10 Students will develop an awareness of career opportunities in the Computer Programming/Software Engineering industry and of its history.

OBJECTIVES

110201-1001 Develop career awareness related to working in the Computer Programming/Software Engineering industry.
- Identify personal interests and abilities related to Computer Programming/Software Engineering careers
  - Identify personal creative talents

- Identify technical/programming talents
- Identify organizational and leadership skills
- Explore aptitude for innovation
- Determine aptitude for working as a member of a Computer Programming/Software Engineering team

- Identify Computer Science career fields
  - Software Engineer
  - Systems Analyst
  - Applications Programmer (Gaming, Multimedia Etc.)

- Investigate career opportunities, trends, and requirements related to Computer Programming/Software Engineering careers

  - Identify the members of a Computer Programming/Software Engineering team: Team Leader, Analyst, Sr. Developer, Jr. Developer, and Client/Subject Matter Expert
  - Describe work performed by each member of the Computer Programming/Software Engineering team
  - Investigate trends associated with Computer Programming/Software Engineering careers
  - Develop a realistic Student Education Occupation Plan (SEOP) to help guide further educational pursuits

- Identify factors for employability and advancement in Computer Programming/Software Engineering careers
  - Survey existing Computer Programming/Software Engineering businesses to determine what training is required
  - Survey universities and colleges to determine higher education options
  - Develop employability competencies/characteristics: responsibility, dependability, respect, and cooperation
  - Achieve high standards of personal performance
  - Develop a positive work ethic
  - Compile a portfolio of the individual and group programs developed during the course

- Discuss relevant history of computer technology and the Computer Programming/Software Engineering industry.

UTAH ATE SKILL CERTIFICATION

STUDENT PERFORMANCE EVALUATION

Test Number: #802 Test Name: Computer Programming I

(PRINT) Student's Name: _____ Date: _____

(PRINT) Teacher's Name: _____ School: _____

Teacher's Signature: _____ District: _____

The performance evaluation **is a required component of the skill certification process**. Each student must be evaluated on the required performance objectives below. Performance objectives may be completed and evaluated anytime during the course. Students who achieve a 3 or 4 (moderately to highly skilled) on **ALL** performance objectives, and 80% on the written test will be issued an ATE skill certificate.

INSTRUCTIONS:

- Students should be aware of their progress throughout the course so that they can concentrate on the objectives that need improvement.
- Students should be encouraged to repeat the objectives until they have performed at a minimum of a number **3 or 4 on the rating scale (moderately to highly skilled level)**.

4 = highly skilled	Successfully demonstrated without supervision
3 = moderately skilled	Successfully demonstrated with limited supervision
2 = limited skill	Demonstrated with close supervision
1 = not skilled	Demonstration requires direct instruction and supervision

- When a performance objective has been achieved at a minimum of 80% (moderately to highly skilled level), **"Y" (Y=YES)** is recorded on the performance summary evaluation form. If a student does not achieve a 3 or a 4 (moderately to highly skilled level), then an **"N" (N=NO)** is recorded on the summary sheet for that objective.
  - All performance objectives **MUST** be completed and evaluated prior to the written test.
  - The teacher will bubble in **"A"** on the skill certification answer sheet (SCANTRON) for item **#81** for students who have achieved **"Y"** on **ALL performance objectives**.
  - The teacher will bubble in **"B"** on the skill certification answer sheet (SCANTRON) for item **#81** for students who have **ONE or more "N's"** on the performance objectives.
- The signed evaluation sheet(s) **MUST** be kept in the teacher's file for two years.
- A copy is also kept on file with the school's ATE skills certification testing coordinator for two years.

Computer Programming I
Performance Objectives

Yes		No		
4	3	2	1	**Standard 1 - The student is familiar with and uses a programming environment.**
				☐ Demonstrated the ability to use an operating system. ☐ Demonstrated the ability to enter, edit, compile, debug and execute programs.
Yes		**No**		**Standard 2 - The student employs accepted programming methodology.**
4	3	2	1	
				☐ Demonstrated the ability to use proper programming style, order, and design techniques with automatic error checking.
Yes		**No**		**Standard 3 - The student properly uses language-fundamental commands and operations.**
4	3	2	1	
				☐ Demonstrated the ability to use basic elements of a specific language such as: templates, variables, constants, arithmetic expressions, input, and output.
Yes		**No**		**Standard 4 - The student properly employs control structures.**
4	3	2	1	
				☐ Demonstrated the ability to use relational and logical operators, decisions, loops, recursion and sub-programs
Yes		**No**		**Standard 5 - The student employs proper static data structures.**
4	3	2	1	
				☐ Demonstrated the ability to use atomic data types, static arrays, and strings.
Yes		**No**		**Standard 6 -The student properly employs object-oriented programming techniques.**
4	3	2	1	
				☐ Demonstrated the ability to use classes, including objects, object data members and member methods (functions). ☐ Demonstrated the ability to create and use user-defined classes including user-defined data members and methods..
Yes		**No**		**Standard 7 - The student properly uses sequential files.**
4	3	2	1	
				☐ Demonstrated the ability to create, initialize, update sequential files. ☐ Demonstrated the ability to store and retrieve data from sequential files.
Yes		**No**		**Standard 8 - The student has applied appropriate programming skills as an effective member of a team.**
4	3	2	1	
				☐ Demonstrated the ability to work as an effective member of a development team completing assigned work according to predetermined time lines, with professionalism and a good attitude. ☐ Developed a software application as a member of a team.
Yes		**No**		**Standard 9 - The student has demonstrated knowledge of current ethical issues dealing with computers and information in society.**
4	3	2	1	
				☐ Demonstrated proper ethical behavior in class, and understands the consequences and ramifications of society's dependance on computers in regard to data privacy and security.
Yes		**No**		**Standard 10 - The student has developed an awareness of career opportunities in the computer programming/software engineering industry and of its history.**
4	3	2	1	
				☐ Demonstrated an understanding of the computer programming/software engineering industry and requirements for employment. ☐ Compiled a portfolio of individual and group programs developed during the course.

Computer Programming II

Grade Levels: 10-12
Units of Credit: 1
CIP Code: 11.0202
Prerequisite: Computer Programming I
Skill Certification Exam: #803 and/or AP Computer Science

COURSE DESCRIPTION

An advanced course in computer programming/software engineering and applications. Reviews and builds on the concepts introduced in CEI. Introduces students to dynamic allocation of data, advanced utilization of classes, advanced GUI techniques, and advanced applications of recursion.

CORE STANDARDS, OBJECTIVES, AND INDICATORS

STANDARD
110202-01 Students will develop applications which make advanced use of the skills and concepts developed in Computer Programming I.

OBJECTIVES:
110202-0101 Demonstrate the ability to develop advanced applications.

- Develop advanced applications using input, calculations, and output.
- Develop advanced applications using IF structures.
- Develop advanced applications using iteration.
- Develop advanced applications using sub-programs.
- Develop advanced applications in object-oriented programming.
- Develop advanced applications using recursion.
- Develop advanced applications using arrays.
- Develop advanced application projects.
- Sort arrays using binary (n log n) sorts.
- Develop advanced applications using files (random access files) or a simple database.

STANDARD
110202-02 Students will use more efficient searching and sorting algorithms.

OBJECTIVES::
110202-0201 Demonstrate the ability to search data structures in programs.

- Develop a binary search.

- Develop a hash search including best and worst, average and hash searches.
- Compare efficiency of sequential and binary searches.

110202-0202 Demonstrate the ability to sort data structures in programs.
- Sort arrays using quadratic (n^2) sorts.
- Sort arrays using binary (n log n) sorts.
- Compare the efficiency of the various sorts using BigO notation including best, worst and average.

STANDARD
110202-03 Students will implement and manipulate a simple database.

OBJECTIVES::
110202-0301 Demonstrate the ability to use random access files in programs.
- Create and initialize random access files.
- Store data to random access files.
- Read data from random access files.
- Update random access files.
- Index random access files.
- Utilize hashing on random access files.

STANDARD
110202-04 Students will properly employ dynamic data structures and abstract data types (ADTs).

OBJECTIVES
110202-0401 Demonstrate the ability to use linked lists in programs.
- Declare pointer identifiers.
- Create node identifiers.
- Insert nodes into a linked list (front, middle, end).
- Delete nodes from a linked list (front, middle, end).
- Output the values in a linked list.
- Search for a value in a linked list.
- Use header and non-header linked lists.
- Perform other linked list operations.
- Develop linked list applications.

110202-0402 Demonstrate the ability to use stacks (arrays and linked lists) in programs.
- Declare stack structures.
- Initialize stacks.
- Check for empty and full stacks.

- Push on to and pop off values from stacks.
- Develop stack applications.

110202-0403 Demonstrate the ability to use queues (arrays and linked lists) in programs.
- Declare queue structures.
- Check for empty and full queues.
- Initialize queues.
- Enqueue values on to and dequeue values off of queues.
- Develop queue applications.

110202-0404 Demonstrate the ability to use binary trees in programs.
- Declare pointer identifiers.
- Create binary tree nodes identifiers.
- Insert nodes into a binary tree.
- Delete nodes from a binary tree.
- Output the values in a binary tree.
- Search for values in ordered binary trees.
- Develop binary tree applications.

STANDARD
110202-05 Students will design and implement classes using inheritance and composition.

OBJECTIVES
110202-0501 Create user-defined inherited classes.

110202-0502 Demonstrate overloading techniques.
- Demonstrate function overloading
- Demonstrate operator overloading (C++ only)

STANDARD
110202-06 Students will develop an individual program of significant complexity and size (300-500 lines).

OBJECTIVES
110202-0601 Create an individual program of significant complexity and size (300-500 lines).

110202-0602 Compile a portfolio of the individual and group programs developed during the course.

STANDARD

110202-07 Students will participate in a work-based learning experience and/or competition.

OBJECTIVES

110202-0701 Participate in a work-based learning experience.

- Field trip to a software engineering firm
- Job shadow
- Internship
- Industry guest speaker
- Post-secondary guest speaker
- Industry interview

110202-0702 Participate at a student programming competition.

- University of Utah High School Computer Programming Contest
- Utah State University High School Computer Programming Contest
- Utah Valley State College Technology Fair (robotics)

UTAH ATE SKILL CERTIFICATION

STUDENT PERFORMANCE EVALUATION

Test Number: #803 Test Name: Computer Programming II

(PRINT) Student's Name: _____ Date: _____

(PRINT) Teacher's Name: _____ School: _____

Teacher's Signature: _____ District: _____

The performance evaluation **is a required component of the skill certification process**. Each student must be evaluated on the required performance objectives below. Performance objectives may be completed and evaluated anytime during the course. Students who achieve a 3 or 4 (moderately to highly skilled) on **ALL** performance objectives, and 80% on the written test will be issued an ATE skill certificate.

INSTRUCTIONS:

- Students should be aware of their progress throughout the course so that they can concentrate on the objectives that need improvement.
- Students should be encouraged to repeat the objectives until they have performed at a minimum of a number **3 or 4 on the rating scale (moderately to highly skilled level)**.

4 = highly skilled	Successfully demonstrated without supervision
3 = moderately skilled	Successfully demonstrated with limited supervision
2 = limited skill	Demonstrated with close supervision
1 = not skilled	Demonstration requires direct instruction and supervision

- When a performance objective has been achieved at a minimum of 80% (moderately to highly skilled level), **"Y" (Y=YES)** is recorded on the performance summary evaluation form. If a student does not achieve a 3 or a 4 (moderately to highly skilled level), then an **"N" (N=NO)** is recorded on the summary sheet for that objective.
  - All performance objectives **MUST** be completed and evaluated prior to the written test.
  - The teacher will bubble in **"A"** on the skill certification answer sheet (SCANTRON) for item **#81** for students who have achieved **"Y"** on **ALL performance objectives**.
  - The teacher will bubble in **"B"** on the skill certification answer sheet (SCANTRON) for item **#81** for students who have **ONE or more "N's"** on the performance objectives.
- The signed evaluation sheet(s) **MUST** be kept in the teachers' file for two years.
- A copy is also kept on file with the school's ATE skills certification testing coordinator for two years.

Computer Programming II Performance Objectives				
Yes		No		**Standard 1 - The student has developed applications which make advanced use of the skills and concepts developed in Computer Programming I.**
4	3	2	1	
				☐ Developed advanced applications using input, calculations, output, IF structures, iteration, sub-programs, recursion, arrays, sorting and a database ☐ Developed advanced application projects ☐ Developed advanced applications using object-oriented programming
Yes		No		**Standard 2 - The student has used more efficient searching and sorting algorithms.**
4	3	2	1	
				☐ Demonstrated the ability to search data structures using binary and hash searches comparing the efficiency between sequential and binary searches. ☐ Demonstrated the ability to sort data structures using quadratic (n^2) and binary (n log n) sorts comparing the efficiency between various sorts using BigO notation.
Yes		No		**Standard 3 - The student has implemented and manipulated a simple database.**
4	3	2	1	
				☐ Demonstrated the ability to use random access files in a program
Yes		No		**Standard 4 - The student has properly employed dynamic data structures and abstract data types (ADTs).**
4	3	2	1	
				☐ Demonstrated the ability to use linked lists, stacks, queues and binary trees
Yes		No		**Standard 5 - The student has designed and implemented classes using inheritance and composition.**
4	3	2	1	
				☐ Created user-defined inherited classes demonstrating overloading techniques
Yes		No		**Standard 6 - The student has developed an individual program of significant complexity and size (300-500 lines).**
4	3	2	1	
				☐ Created an individual program of significant complexity and size (300-500 lines). ☐ Compiled a portfolio of the individual and group programs developed during the course.
Yes		No		**Standard 7 - The student has participated in a work-based learning experience and/or competition.**
4	3	2	1	
				☐ Participated in a work-based learning experience such as a job shadow, internship, field trip to a software engineering firm or listened to an industry guest speaker and/or competed in a high school programming contest

Resources

Reference Books:

Java: Complete Course in Programming & Problem Solving, Lambert, Southwestern/Thomson, 2000.
Microsoft Visual Basic Basics, Knowlton, Southwestern/Thomson, 2000
Microsoft Visual Basic 6.0 Introduction to Programming, Sprague, Southwestern/Thomson, 2000.
Microsoft Visual Basic Programming Projects, CEP, Southwestern/Thomson, 2000.
Microsoft Visual Basic, Sprague/Phillips, South Western.
Introduction to Programming, Sprague, Southwestern/Thomson, 2000.
Visual Basic 6.0: Basics, Knolton/Collings, South Western.
Visual Basic Programming Projects, CEP/Sestak, South Western, 2000.

Websites:

www.mainfunction.com/default.asp	MainFunction.com - teacher & student site
www.teach-scheme.org/Workshops/#materials	Teach Scheme Project

See www.usoe.k12.ut.us/ate/it/cool.htm for additional web sites.

Software Compiler Vendors

Inprise (Borland)
100 Enterprise Way
Scotts Valley, CA 95066-3249
(800) 847-7797
www.borland.com/programs/education/baer/baer.html

JCreator (freeware)
www.jcreator.com

Metrowerks (Code Warrior)
2201 Donley Drive, Suite 310
Austin Texas 78758
(512) 873-4700 or (512) 873-4716
www.metrowerks.com

Microsoft
One Microsoft Way
Redmond Washington 98052 - 6399
(800) 426-9400
www.microsoft.com/education/k12/index.htm

PLT Scheme (Freeware)
www.plt-scheme.org/

Sun Microsystems
901 San Antonio Road
Palo Alto, CA 94303
(650) 960-1300
www.sun.com/aboutsun/coinfo/sunorg.html
java.sun.com

Hardware (Equipment) and Software (Programs) Requirements

Minimal: l Windows environment workstation per student
- Pentium Processor
- Windows 98 or Better
- 64 Megabytes of RAM
- CD-ROM drive for installation (could be external)
- 200 Megabytes of free hard drive space (check software specifications)

or 1 Mac OS workstation per student
- PowerPC or higher
- System 8 or Better
- 64 Megabytes of RAM
- CD-ROM drive for installation (could be external)
- 200 Megabytes of free hard drive space (check software specifications)

1 license (copy) of a current language compiler for each workstation
- Could be C++, Java, Visual Basic

Optimal: l Windows environment workstation per student attached to the Internet
- Pentium IV processor or higher
- Windows 2000 or higher
- 256 Megabytes of RAM
- CD-ROM drive for installation
- 20 G of free hard drive space

or 1 Mac OS workstation per student attached to the Internet
- G4 or higher
- System X or later
- 256 Megabytes of RAM
- CD-ROM drive for installation (could be external)
- 20 G of free hard drive space

1 license (copy) of a current language compiler for each workstation
- Could be C++, Java, Visual Basic

(or state-of-the-art, if higher than above)

Glossary

Atomic Data Type: Indivisible. An atomic operation, or atomicity, implies an operation that must be performed entirely or not at all. For example, if machine failure prevents a transaction to be processed to completion, the system will be rolled back to the start of the transaction.

Binary Number: Numbers stored in pure binary form. Within one byte (8 bits), the values 0 to 255 can be held. Two contiguous bytes (16 bits) can hold values from 0 to 65,535.

Bottom-up Design:

Compile: To translate a program written in a high-level programming language into machine language.

Computer Software: Instructions for the computer. A series of instructions that performs a particular task is called a "program." The two major categories of software are "system software" and "application software." System software is made up of control programs such as the operating system and database management system (DBMS). Application software is any program that processes data for the user (inventory, payroll, spreadsheet, word processor, etc.).

Debug: To correct a problem in hardware or software. Debugging software is finding the errors in the program logic.

Documentation: The narrative and graphical description of a system. System documentation includes:

- Data dictionary - Description of the files and databases.
- System flow chart - Description of the data as it flows from source document to report.
- Application program documentation - Description of the inputs, processing and outputs for each data entry, query, update and report program in the system.

Encapsulation: In object technology, making the data and processing within the object private, which allows the internal implementation of the object to be modified without requiring any change to the application that uses it.

Ethics: Rules of personal behavior accepted by society. Various professional organizations create codes of ethics for members of their group.

Logic Error: A program bug due to an incorrect sequence of instructions.

Object: A self-contained module of data and its associated processing. Objects are the software building blocks of object technology.

Object-Oriented Programming (OOP): Abbreviated "OOP," programming that supports object

technology. It is an evolutionary form of modular programming with more formal rules that allow pieces of software to be reused and interchanged between programs. Major concepts are encapsulation, inheritance and polymorphism.

Polymorphism: In object technology, the ability of a generalized request (message) to produce different results based on the object that it is sent to.

Programming Languages: A language used to write instructions for the computer. Like human languages, each programming language has its own grammar and syntax. Programming languages fall into two categories: low-level assembly languages and high-level languages. Assembly languages are available for each CPU family, and each assembly instruction is translated into one machine instruction by the assembler program. With high-level languages, a programming statement may be translated into one or several machine instructions by the compiler.

Robustness: Refers to software without bugs that handles abnormal conditions well. It is often said that there is no software package totally bug free. Any program can exhibit odd behavior under certain conditions, but a robust program will not lock up the computer, cause damage to data or send the user through an endless chain of dialog boxes without purpose.

Run-time Error: A problem that is encountered when a program is being executed.

Syntax: The rules governing the structure of a language statement.

Syntax Error: An error that occurs when a program cannot understand the command that has been entered.

System/Software Development Life Cycle: The sequence of events in the development of an information system (application), which requires mutual effort on the part of user and technical staff.

Topdown Design: A design technique that starts with the highest level of an idea and works its way down to the lowest level of detail.